Map Task Scheduling in MapReduce with Data Locality: Throughput and Heavy-Traffic Optimality

Weina Wang, Kai Zhu and Lei Ying
Electrical, Computer and Energy Engineering
Arizona State University
Tempe, Arizona 85287
{weina.wang, kzhu17, Lei.Ying.2}@asu.edu

Jian Tan and Li Zhang
IBM T. J. Watson Research Center
Yorktown Heights, New York, 10598
{tanji, zhangli}@us.ibm.com

***Abstract*—Scheduling map tasks to improve data locality is crucial to the performance of MapReduce. Many works have been devoted to increasing data locality for better efficiency. However, to the best of our knowledge, fundamental limits of MapReduce computing clusters with data locality, including the capacity region and theoretical bounds on the delay performance, have not been studied. In this paper, we address these problems from a stochastic network perspective. Our focus is to strike the right balance between data-locality and load-balancing to simultaneously maximize throughput and minimize delay. We present a new queueing architecture and propose a map task scheduling algorithm constituted by the Join the Shortest Queue policy together with the MaxWeight policy. We identify an outer bound on the capacity region, and then prove that the proposed algorithm stabilizes any arrival rate vector strictly within this outer bound. It shows that the algorithm is throughput optimal and the outer bound coincides with the actual capacity region. Further, we study the number of backlogged tasks under the proposed algorithm, which is directly related to the delay performance based on Little's law. We prove that the proposed algorithm is heavy-traffic optimal, i.e., it asymptotically minimizes the number of backlogged tasks as the arrival rate vector approaches the boundary of the capacity region. Therefore, the proposed algorithm is also delay optimal in the heavy-traffic regime.**

I. Introduction

Processing large-scale datasets has become an increasingly important and challenging problem as the amount of data created by online social networks, healthcare industry, scientific research, etc., explodes. MapReduce/Hadoop [1, 2] is a simple yet powerful framework for processing large-scale datasets in a distributed and parallel fashion, and has been widely used in practice, including Google, Yahoo!, Facebook, Amazon and IBM.

A production MapReduce cluster may even consist of tens of thousands of machines [3]. The stored data are typically organized on distributed file systems (e.g., Google File System (GFS) [4], Hadoop Distributed File System (HDFS) [5]), which divide a large dataset into data chunks (e.g., 64 MB) and store multiple replicas (by default 3) of each chunk on different machines. A data processing request under the MapReduce framework, called a job, consists of two types of tasks: *map* and *reduce*. A map task reads one data chunk and processes it to produce intermediate results (key-value pairs). Then reduce tasks fetch the intermediate results and carry out further computations to produce the final result. Map and reduce tasks are assigned to the machines in the computing cluster by a master node which keeps track of the status of these tasks to manage the computation process. In assigning map tasks, a critical consideration is to place map tasks on or close to machines that store the input data chunks, a problem called *data locality*.

For each task, we call a machine a *local machine* for the task if the data chunk associated with the task is stored locally, and we call this task a *local task* on the machine; otherwise, the machine is called a *remote machine* for the task and correspondingly this task is called a *remote task* on the machine. The term *locality* is also used to refer to the fraction of tasks that run on local machines. Improving locality can reduce both the processing time of map tasks and the network traffic load since fewer map tasks need to fetch data remotely. However, assigning all tasks to local machines may lead to an uneven distribution of tasks among machines, i.e., some machines may be heavily congested while others may be idle. Therefore, we need to strike the right balance between data-locality and load-balancing in MapReduce.

In this paper, we call the algorithm that assigns map tasks to machines a map-scheduling algorithm or simply a scheduling algorithm. There have been several attempts to increase data locality in MapReduce to improve the system efficiency. For example, the currently used scheduling algorithms in Google's MapReduce and Hadoop take the location information of data chunks into account and attempt to schedule a map task as close as possible to the machine that has the data chunk [1, 6, 7]. A scheduling algorithm called *delay scheduling*, which delays some tasks for a small amount of time to attain higher locality, has been proposed in [7]. In addition to scheduling algorithms, data replication algorithms such as Scarlett [3] and DARE [8] have also been proposed.

While the data locality issue has received a lot of attention and scheduling algorithms that improve data locality have been proposed in the literature and implemented in practice, to the best of our knowledge, none of the existing works have studied the fundamental limits of MapReduce computing clusters with data locality. Basic questions such as *what is the capacity* *region of a MapReduce computing cluster with data locality*, *which scheduling algorithm can achieve the full capacity region*, *how to minimize the waiting time and congestion in a MapReduce computing cluster with data locality*, remain open.

In this paper, we will address these basic questions from a stochastic network perspective. Motivated by the observation that a large portion of jobs are map-intensive, and many of them only require map tasks [9], we focus on map scheduling algorithms and assume reduce tasks are either not required or not the bottleneck of the job processing. We assume that the data have been divided into chunks, and each chunk has three replicas stored on three different machines. The computing cluster is modeled as a time-slotted system, in which jobs consisting of a number of map tasks arrive at the beginning of each time slot according to some stochastic process. Each map task processes one data chunk and map tasks are *nonpreemptive*. Within each time slot, a task is completed with probability α at a local machine, or with probability γ ($\gamma < \alpha$) at a remote machine, i.e., the service times are geometrically distributed with different parameters. Based on this model, we establish the following fundamental results:

- First, we present an outer bound on the capacity region of a MapReduce computing cluster with data locality, where the capacity region consists of all arrival vectors for which there exists a scheduling algorithm that stabilizes the system (stability region in [10]).
- We propose a new queueing architecture with one local queue for each machine, storing local tasks associated with the machine, and a common queue for all machines. Based on this new queueing architecture, we propose a two-stage map scheduling algorithm under which a newly arrived task is routed to one of the three queues associated with the three local machines or the common queue using the Join the Shortest Queue (JSQ) policy; and when a machine is idle, it selects a task from the local queue associated with it or the common queue using the MaxWeight policy [10].
- We prove that the joint JSQ and MaxWeight scheduling algorithm is throughput optimal, i.e., it can stabilize any arrival rate vector strictly within the outer bound of the capacity region, which also shows that the outer bound is tight and is the same as the actual capacity region. We remark that existing results on MaxWeight-based scheduling algorithms assume deterministic processing (service) time or geometrically distributed processing time with preemptive tasks. To the best of our knowledge, the stability of MaxWeight scheduling with random processing time and nonpreemptive tasks has not been established before. So the proof technique itself is a novel contribution of this paper, and may be extended to prove the stability of MaxWeight scheduling for other applications, in which the service times are geometrically distributed. We remark that recently in [11], the authors studied MaxWeight scheduling for resource allocation in clouds and independently established a similar result with more general service time distributions.
- In addition to throughput optimality, we further study the number of backlogged tasks, which is directly related to the delay performance based on Little's law. We prove that under a heavy local traffic condition, the joint JSQ and MaxWeight scheduling algorithm is heavy-traffic optimal, i.e., it asymptotically minimizes the number of backlogged tasks as the arrival rate vector approaches the boundary of the capacity region. Therefore, the proposed algorithm strikes the right balance between data-locality and load-balancing and is both throughput and delay optimal in the heavy-traffic regime. The proof of heavy-traffic optimality follows the Lyapunov drift analysis recently developed in [12]. Heavy-traffic optimality of JSQ only and MaxWeight only have been proved in [12]. In [13], the result has been extended to a joint JSQ and MaxWeight algorithm (in a different context) when servers are homogeneous. In this paper, the machines are heterogeneous due to data locality. The proof of heavy-traffic optimality is a non-trivial application of the drift-based analysis.

II. System Model

We consider a discrete-time model for a computing cluster with M machines, indexed $1, 2, \cdots, M$. Jobs come in stochastically and when a job comes in, it brings a random number of map tasks, which need to be served by the machines. We assume that each data chunk is replicated and placed at three different machines. Therefore, each task is associated with three local machines. It takes longer time for a machine to process a task if the required data chunk is not stored locally since the machine needs to retrieve the data first. Tasks can be classified according to the local machines they associate with. For each task, we assemble the indices of its three local machines in an increasing order into a vector

$$\vec{L} \in \left\{(m_1, m_2, m_3) \in \{1, 2, \cdots, M\}^3, m_1 < m_2 < m_3\right\},$$

which forms the *type* of the task. The notation $m \in \vec{L}$ indicates that machine m is a local machine for type $\vec{L}$ tasks. Let $\mathcal{L}$ denote the set of the existing types in the cluster and $N = |\mathcal{L}|$.

A. Arrival and Service

Let $A_{\vec{L}}(t)$ denote the number of type $\vec{L}$ tasks arriving at the beginning of time slot t. We assume that the arrival process is temporally i.i.d. with arrival rate $\lambda_{\vec{L}}$. We further assume the arrival processes are bounded. At each machine, the service times of tasks follow geometric distributions. The parameter of the geometric distribution is α for a task at a local machine, and γ at a remote machine. The service process of a task can be viewed as a sequence of independent trials with success probability α or γ, and the sequence stops once we get a success, i.e., once the task is finished. In this model, we assume the parameters satisfy $\alpha > \gamma$. Then the average service time of local tasks is less than that of remote tasks; i.e., $1/\alpha < 1/\gamma$. Note that α and γ characterize the different processing efficiency due to data locality.

B. Task Scheduling Algorithm

The task scheduling problem is to assign incoming tasks to the machines. Due to data locality, the task scheduling algorithm can significantly affect the efficiency of the system. In this paper, we consider a task scheduling algorithm consisting of two parts: routing and scheduling. We present a new queue architecture as illustrated in Fig. 1. The master node maintains a queue for each machine m for local tasks, denoted by Q_m and called the *local queue*; and there is a common queue for all machines, denoted by $\overline{Q}$ (or sometimes Q_{M+1}), and called the *common remote queue*. We use a queue length vector $Q(t) = \big(Q_1(t), \cdots, Q_M(t), \overline{Q}(t)\big)$ to denote the queue lengths at the beginning of time slot t. When a task comes in, the master node routes the task to some queue in the queueing system. When a machine is idle, it picks a task from the corresponding local queue or the common remote queue to serve. These two steps are illustrated in Fig. 1. We call the first step *routing*, and with a slight abuse of terminology we call the second step *scheduling*. It should be clear from the context that whether we are referring to the whole task scheduling problem or to this service scheduling step. Based on our queue architecture, we propose the following joint routing and scheduling algorithm.

- **Join the Shortest Queue (JSQ) Routing.** When a task comes in, the master node compares the queue lengths of its three local queues and the common remote queue, and then routes the task to the shortest one. Ties are broken randomly. Let $A_{\vec{L},m}(t)$ and $\overline{A}_{\vec{L}}(t)$ denote the arrivals of type $\vec{L}$ tasks allocated to Q_m and $\overline{Q}$, respectively. Then the arrivals allocated to each queue can be expressed by the arrival vector $A(t) = \big(A_1(t), \cdots, A_M(t), \overline{A}(t)\big)$, defined as

$$\begin{aligned} A_m(t) &= \textstyle\sum_{\vec{L}:\, m\in\vec{L}} A_{\vec{L},m}(t),\ m = 1, 2, \cdots, M \\ \overline{A}(t) &= \textstyle\sum_{\vec{L}} \overline{A}_{\vec{L}}(t). \end{aligned}$$

- **MaxWeight Scheduling.** If machine m just finished a task at time slot $t-1$, then its working status is *idle*. Otherwise, the machine must be working on some local or remote task. Let $f_m(t) = 0, 1, 2$ denote idle, working on a local task, and working on a remote task, respectively. The working status vector $f(t) = (f_1(t), f_2(t), \cdots, f_m(t))$ and queue length vector $Q(t)$ are reported to the master at the beginning of time slot t, and the master makes scheduling decisions for all the machines based on $f(t)$ and $Q(t)$. The idle machines are scheduled according to the MaxWeight algorithm: suppose machine m is idle at time slot t, then it serves a local task if $\alpha Q_m(t) \geq \gamma\overline{Q}(t)$ and a remote task otherwise. Other machines continue to serve the unfinished tasks, i.e., the execution of tasks is *non-preemptive*. Let $\sigma_m(t)$ denote the scheduling decision of machine m at time slot t, then it is a function of $Q(t)$ and $f_m(t)$, and

$$\sigma_m(t) = \begin{cases} 1 & \text{if a local task is to be served,} \\ 2 & \text{if a remote task is to be served.} \end{cases}$$

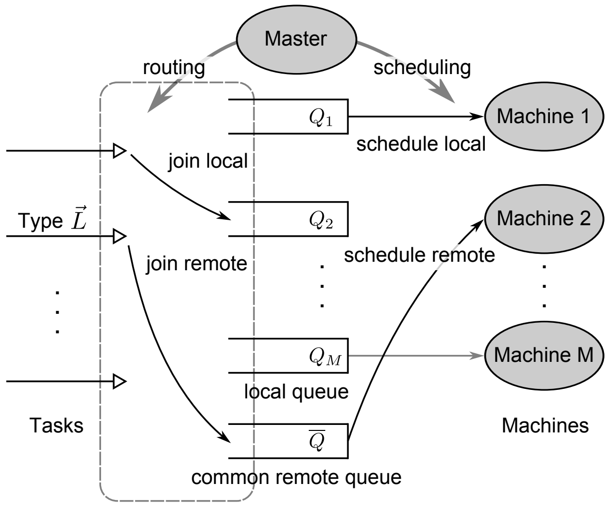


Fig. 1: The Queue Architecture and Scheduling Algorithm

Note that $\sigma_m(t)$ indicates which queue machine m is scheduled to serve. It can only take value 1 or 2 since the machine is scheduled to serve either a local task or a remote task. If machine m is not idle, i.e., $f_m(t) = 1$ or 2, the schedule $\sigma_m(t)$ equals to $f_m(t)$ by our settings. However if machine m is idle, i.e., $f_m(t) = 0$, $\sigma_m(t)$ is still either 1 or 2, decided by the master according to the MaxWeight algorithm. We use the schedule vector $\sigma(t) = (\sigma_1(t), \sigma_2(t), \cdots, \sigma_M(t))$ to denote the scheduling decisions of all the machines.

Here we note that each queue in this architecture can actually be divided into multiple subqueues according to the job that the task comes from, i.e., per job subqueues. Then in the scheduling step, an idle machine can further pick a subqueue to serve for the fairness purpose. However, this change will not affect our analysis throughout this paper, so we only consider this structure in the simulation part.

C. Queue Dynamics

In time slot t, first the master checks the working status information $f(t)$ and the queue length $Q(t)$. Then the tasks arrive at the master and the master does the routing and the scheduling, yielding $A(t)$ and $\sigma(t)$. Define

$$\begin{cases} \mu_m^l(t) = \alpha,\ \mu_m^r(t) = 0 & \text{if } \sigma_m(t) = 1, \\ \mu_m^l(t) = 0,\ \mu_m^r(t) = \gamma & \text{if } \sigma_m(t) = 2. \end{cases}$$

The service from machine m to local queue Q_m and remote queue $\overline{Q}$ are two Bernoulli random variables $S_m^l(t) \sim \text{Bern}\big(\mu_m^l(t)\big)$ and $S_m^r(t) \sim \text{Bern}\,(\mu_m^r(t))$. Hence the service applied to each queue can be expressed by the service vector $S(t) = \Big(S_1^l(t), \cdots, S_M^l(t), \sum_{m=1}^M S_m^r(t)\Big)$, which is the service process we introduced in Section II-A with service rate α or γ. Then the queue lengths satisfy the following equations.

- **Local queues.** For any $m = 1, 2, \cdots, M$,

$$Q_m(t+1) = Q_m(t) + A_m(t) - S_m^l(t) + U_m(t),$$

where

$$U_m(t) = \begin{cases} 0 & \text{if } Q_m(t) + A_m(t) \geq 1, \\ S^l_m(t) & \text{if } Q_m(t) + A_m(t) = 0. \end{cases}$$

- **The Remote queue.**

$$\overline{Q}(t+1) = \overline{Q}(t) + \overline{A}(t) - \textstyle\sum_{m=1}^{M} S^r_m(t) + \overline{U}(t),$$

where

$$\overline{U}(t) = \textstyle\sum_{m=1}^{M} S^r_m(t) - \sum_{m \in \mathcal{A}(t)} S^r_m(t)$$

and $\mathcal{A}(t)$ is the set of machines which actually have some tasks to serve from the remote queue at time slot t. Note that there can be some machines which attempt to serve the remote queue but fail due to insufficient tasks.

By our notations, the queue dynamics can thus be written as

$$Q(t+1) = Q(t) + A(t) - S(t) + U(t), \tag{1}$$

where $U(t) = \big(U_1(t), \cdots, U_M(t), \overline{U}(t)\big)$ is the unused service.

In the case that the service time is deterministic, the queueing process $\{Q(t), t \geq 0\}$ itself is a Markov chain. However, the service time in our model is random and heterogeneous due to data locality. Thus we need to also consider the working status vector $f(t)$, and $Q(t)$ together with $f(t)$ forms a Markov chain $\{(Q(t), f(t)), t \geq 0\}$. We assume the initial state is $\big(Q(0), f(0)\big) = \big(0_{(M+1)\times 1}, 0_{M \times 1}\big)$ and the state space $\mathcal{S} \subseteq \mathbb{N}^{M+1} \times \{0,1,2\}^M$ consists of all the states which can be reached from the initial state, where $\mathbb{N}$ is the set of nonnegative integers. Then this Markov chain is irreducible and aperiodic.

III. Throughput Optimality

In this section, we first identify an outer bound of the capacity region of the system. We then prove that the proposed task scheduling algorithm stabilizes any arrival rate vector strictly within this outer bound, which means that the proposed algorithm is *throughput optimal*, and the capacity region coincides with the outer bound.

A. Capacity Region

For any task type $\vec{L} \in \mathcal{L}$, we assume that the number of type $\vec{L}$ arrivals allocated to machine m has a rate $\lambda_{\vec{L},m}$, then $\lambda_{\vec{L}} = \sum_{m=1}^{M} \lambda_{\vec{L},m}$. The set of rates $\{\lambda_{\vec{L},m}\}_{\vec{L}\in\mathcal{L}, m=1,\cdots,M}$ will be called a *decomposition* of the arrival rate vector $\lambda = \big(\lambda_{\vec{L}_1}, \lambda_{\vec{L}_2}, \cdots, \lambda_{\vec{L}_N}\big)$ in the rest of this paper, and the index range may be omitted for conciseness. For any machine m, a necessary condition for an arrival rate vector λ to be supportable is that the average arrivals allocated to machine m in one time slot can be served within one time slot, i.e.,

$$\sum_{\vec{L}:\, m \in \vec{L}} \frac{\lambda_{\vec{L},m}}{\alpha} + \sum_{\vec{L}:\, m \notin \vec{L}} \frac{\lambda_{\vec{L},m}}{\gamma} \leq 1, \tag{2}$$

where the left hand side is the time machine m needs to serve the arrivals allocated to it in one time slot on average, since the service rate is α for local tasks and γ for remote tasks.

Let Λ be the set of arrival rates such that each element has a decomposition satisfying (2). Formally,

$$\begin{aligned} \Lambda = \Bigg\{ & \lambda = \big(\lambda_{\vec{L}_1}, \lambda_{\vec{L}_2}, \cdots, \lambda_{\vec{L}_N}\big) \colon \\ & \lambda_{\vec{L}} = \sum_{m=1}^{M} \lambda_{\vec{L},m},\ \forall \vec{L} \in \mathcal{L}, \\ & \lambda_{\vec{L},m} \geq 0,\ \forall \vec{L} \in \mathcal{L},\ \forall m = 1, \cdots, M \\ & \sum_{\vec{L}:\, m \in \vec{L}} \frac{\lambda_{\vec{L},m}}{\alpha} + \sum_{\vec{L}:\, m \notin \vec{L}} \frac{\lambda_{\vec{L},m}}{\gamma} \leq 1,\ \forall m = 1, \cdots, M \Bigg\}. \end{aligned} \tag{3}$$

Then Λ gives an outer bound of the capacity region.

B. Achievability

Theorem 1 (Throughput Optimality)**.** *The proposed map-scheduling algorithm stabilizes any arrival rate vector strictly within Λ. Hence, this algorithm is throughput optimal, and Λ is the capacity region of the system.*

We give the outline of the proof below due to space limit, and refer to our technical report [14] for the complete proof.

Proof Outline: Since $\{(Q(t), f(t)), t \geq 0\}$ is an irreducible and aperiodic Markov chain, the stability is defined to be the positive recurrence of this Markov chain. By the extension of the Foster-Lyapunov theorem, it is sufficient to find a positive integer T and a Lyapunov function whose T time slot Lyapunov drift is bounded within a finite subset of the state space and negative outside this subset.

Step 1. Consider the Lyapunov function

$$W(Q(t), f(t)) = \|Q(t)\|^2 = \textstyle\sum_{m=1}^{M} Q^2_m(t) + \overline{Q}^2(t).$$

The Lyapunov drift from time slot t_0 to $t_0 + T$ is bounded by

$$2\mathbb{E}\Big[\textstyle\sum_{t=t_0}^{t_0+T-1} \langle Q(t), A(t) - S(t)\rangle \;\big|\; Q(t_0), f(t_0)\Big] + \text{const.} \tag{4}$$

Step 2. Consider an arrival process with arrival rate vector $\lambda = \big(\lambda_{\vec{L}_1}, \cdots, \lambda_{\vec{L}_N}\big) \in \Lambda^o$ and define $\tilde{\lambda} = \big(\tilde{\lambda}_1, \cdots, \tilde{\lambda}_{M+1}\big)$ as

$$\begin{aligned} \tilde{\lambda}_m &= \textstyle\sum_{\vec{L}:\, m \in \vec{L}} \lambda_{\vec{L},m},\ m = 1, 2, \cdots, M, \\ \tilde{\lambda}_{M+1} &= \textstyle\sum_{m=1}^{M} \sum_{\vec{L}:\, m \notin \vec{L}} \lambda_{\vec{L},m}. \end{aligned}$$

Then we can write the expectation in bound (4) as

$$\begin{aligned} & \mathbb{E}\Big[\textstyle\sum_{t=t_0}^{t_0+T-1} \langle Q(t), A(t) - S(t)\rangle \;\big|\; Q(t_0), f(t_0)\Big] \\ &= \mathbb{E}\Big[\textstyle\sum_t \big(\langle Q(t), A(t)\rangle - \langle Q(t), \tilde{\lambda}\rangle\big) \;\big|\; Q(t_0), f(t_0)\Big] \qquad (5) \\ &\quad + \mathbb{E}\Big[\textstyle\sum_t \big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\big) \;\big|\; Q(t_0), f(t_0)\Big]. \qquad (6) \end{aligned}$$

Step 3. The arrival part (5) under the JSQ routing is bounded by Lemma 8 in our technical report [14] as,

$$\mathbb{E}\Big[\textstyle\sum_t \big(\langle Q(t), A(t)\rangle - \langle Q(t), \tilde{\lambda}\rangle\big) \;\big|\; Q(t_0), f(t_0)\Big] \leq 0.$$

Step 4. To bound the service part (6), we start with the following random variables

$$\begin{aligned} t^*_m &= \min\{\tau \colon \tau \geq t_0, f_m(\tau) = 0\},\ m = 1, 2, \cdots, M, \\ t^* &= \max_{1 \leq m \leq M} t^*_m. \end{aligned} \tag{7}$$

Thus t^*_m is the first time slot after t_0 at which machine m is idle and makes a scheduling decision, and t^* is the first time slot by which every machine has been idle for at least once since t_0. We use t^* to decompose the probability space. Let $T = JK$, then

$$
\begin{aligned}
&\mathbb{E}\Big[\textstyle\sum_t \Big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\Big) \mid Q(t_0), f(t_0)\Big] \\
&= \mathbb{E}\Big[\textstyle\sum_t \Big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\Big) \mid Q(t_0), f(t_0), \qquad (8)\\
&\quad t^* \geq t_0 + K\Big] \cdot \Pr\left(t^* \geq t_0 + K \mid Q(t_0), f(t_0)\right) \\
&+ \mathbb{E}\Big[\textstyle\sum_t \Big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\Big) \mid Q(t_0), f(t_0), \qquad (9)\\
&\quad t^* < t_0 + K\Big] \cdot \Pr\left(t^* < t_0 + K \mid Q(t_0), f(t_0)\right).
\end{aligned}
$$

Step 4a. For term (8), by boundedness of arrival and service,

$$
\begin{aligned}
&\mathbb{E}\Big[\textstyle\sum_t \Big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\Big) \,|\, Q(t_0), f(t_0), t^* \geq t_0 + K\Big] \\
&\leq TMQ_\Sigma(t_0) + \text{const},
\end{aligned}
$$

where $Q_\Sigma = \sum_{m=1}^M Q_m + \overline{Q}$ denote the queue length sum.

Step 4b. For term (9), we further condition on t^*. For the summation from $t = t_0$ to t^*, under the condition $t^* < t_0 + K$,

$$
\begin{aligned}
&\textstyle\sum_{t=t_0}^{t^*} \mathbb{E}\Big[\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle \mid t^*, Q(t_0), f(t_0)\Big] \\
&\leq KMQ_\Sigma(t_0) + \text{const}.
\end{aligned}
$$

For the summation from $t = t^* + 1$ to $t_0 + T - 1$, first since $\lambda \in \Lambda^o$, there exists $\epsilon > 0$ and a decomposition $\left\{\lambda_{\vec{L},m}\right\}$ such that

$$
\sum_{\vec{L}:\, m \in \vec{L}} \frac{\lambda_{\vec{L},m}}{\alpha} + \sum_{\vec{L}:\, m \notin \vec{L}} \frac{\lambda_{\vec{L},m}}{\gamma} \leq \frac{1}{1+\epsilon}. \qquad (10)
$$

Thus

$$
\begin{aligned}
&Q_m(t) \textstyle\sum_{\vec{L}:\, m \in \vec{L}} \lambda_{\vec{L},m} + \overline{Q}(t) \sum_{\vec{L}:\, m \notin \vec{L}} \lambda_{\vec{L},m} \\
&\leq \frac{1}{1+\epsilon} \max\left\{\alpha Q_m(t), \gamma \overline{Q}(t)\right\}.
\end{aligned}
$$

Next consider the random variable τ^t_m defined as

$$
\tau^t_m = \max\left\{\tau \colon \tau \leq t, f_m(\tau) = 0\right\},\ m = 1, 2, \cdots, M, \qquad (11)
$$

which is the last time slot before t at which machine m is idle and is scheduled to some queue based on the MaxWeight algorithm. Thus for each t such that $t^* < t \leq t_0 + T$, we have $t_0 \leq \tau^t_m \leq t_0 + T$. By the MaxWeight algorithm,

$$
\begin{aligned}
&Q_m(\tau^t_m)\mathbb{E}\left[S^l_m(t) \mid \sigma_m(\tau^t_m)\right] + \overline{Q}(\tau^t_m)\mathbb{E}\left[S^r_m(t) \mid \sigma_m(\tau^t_m)\right] \\
&= \begin{cases} \alpha Q_m(\tau^t_m) & \text{if } \sigma_m(\tau^t_m) = 1, \text{ i.e., if } \alpha Q_m(\tau^t_m) \geq \gamma \overline{Q}(\tau^t_m) \\ \gamma \overline{Q}(\tau^t_m) & \text{if } \sigma_m(\tau^t_m) = 2, \text{ i.e., if } \alpha Q_m(\tau^t_m) \leq \gamma \overline{Q}(\tau^t_m) \end{cases} \\
&\geq \max\left\{\alpha Q_m(\tau^t_m), \gamma \overline{Q}(\tau^t_m)\right\}.
\end{aligned}
$$

Then utilizing conditional expectations and the bounded difference between any two of $Q(\tau^t_m)$, $Q(t)$ and $Q(t_0)$, yields

$$
\begin{aligned}
&\textstyle\sum_{t=t^*+1}^{t_0+T} \mathbb{E}\left[\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle \,\Big|\, t^*, Q(t_0), f(t_0)\right] \\
&\leq -\frac{(J-1)K\epsilon\alpha\gamma M}{(1+\epsilon)(\gamma M + \alpha)} Q_\Sigma(t_0) + \text{const}.
\end{aligned}
$$

Combining the two summations gives

$$
\begin{aligned}
&\mathbb{E}\Big[\textstyle\sum_t \Big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\Big) \,|\, Q(t_0), f(t_0), t^* < t_0 + K\Big] \\
&\leq KM\left(1 - \frac{(J-1)\epsilon\alpha\gamma}{(1+\epsilon)(\gamma M + \alpha)}\right) Q_\Sigma(t_0) + \text{const}.
\end{aligned}
$$

Step 4c. We show that $\Pr\left(t^* \geq t_0 + K \mid Q(t_0), f(t_0)\right) \to 0$ as $K \to \infty$ in Lemma 10 of our technical report [14]. Choose large enough K and J, i.e., large enough T, then for some $\theta > 0$,

$$
\begin{aligned}
&\mathbb{E}\Big[\textstyle\sum_t \big(\langle Q(t), \tilde{\lambda}\rangle - \langle Q(t), S(t)\rangle\big) \mid Q(t_0), f(t_0)\Big] \\
&\leq -\theta Q_\Sigma(t_0) + \text{const}.
\end{aligned}
$$

Step 5. The T time slot Lyapunov drift from t_0 is thus bounded as $D\left(Q(t_0), f(t_0)\right) \leq -2\theta Q_\Sigma(t_0) + B$ for a constant $B > 0$.

Step 6. Let $\mathcal{B} = \left\{(Q, f) \in \mathcal{S} \colon Q_1 + \cdots + Q_{M+1} \leq \frac{B+\delta}{2\theta}\right\}$ for an arbitrary $\delta > 0$. Then $\mathcal{B}$ is a finite subset of $\mathcal{S}$ satisfying that for any $(Q, f) \in \mathcal{B}^c$, $D(Q, f) \leq -\delta$ and for any $(Q, f) \in \mathcal{B}$, $D(Q, f) \leq B$. This finishes the proof for stability. Thus the proposed task scheduling algorithm is throughput optimal, and Λ is the capacity region of the system. ■

IV. Heavy-Traffic Optimality

In this section, we analyze the performance of the proposed algorithm beyond throughput. We will show that in the *heavy-traffic* regime, the proposed algorithm asymptotically minimizes the number of backlogged tasks.

Suppose the set of existing task types $\mathcal{L}$ is such that there are M_l machines, each of which is considered as a local machine by some task type, and the other $M_r = M - M_l$ machines are remote machines for all the task types. Denote the set of the machines which can have local tasks as $\mathcal{M}_l$ and the set of the machines which only have remote tasks as $\mathcal{M}_r$. Then

$$
\begin{aligned}
&\mathcal{M}_l = \left\{m \in \{1, 2, \cdots, M\} \colon \exists \vec{L} \in \mathcal{L},\ \text{s.t. } m \in \vec{L}\right\}, \\
&\mathcal{M}_r = \{1, 2, \cdots, M\} - \mathcal{M}_l,
\end{aligned}
$$

and $|\mathcal{M}_l| = M_l$, $|\mathcal{M}_r| = M_r$. Without loss of generality, we assume $\mathcal{M}_l = \{1, \cdots, M_l\}$ and $\mathcal{M}_r = \{M_l + 1, \cdots, M\}$.

We consider the heavy-traffic regime that the arrival rates satisfy that for any subset $\mathcal{H}$ of $\mathcal{M}_l$, the sum arrival rate of local tasks to the machines in $\mathcal{H}$ is larger than the process capacity of the machines. Formally, let $\lambda = \left(\lambda_{\vec{L}_1}, \cdots, \lambda_{\vec{L}_N}\right) \in \Lambda^o$ be the arrival rate vector, then we assume for any $\mathcal{H} \subseteq \mathcal{M}_l$,

$$
\sum_{\vec{L}:\, \exists m \in \mathcal{H},\, \text{s.t. } m \in \vec{L}} \lambda_{\vec{L}} \geq |\mathcal{H}|\alpha, \qquad (12)
$$

which is referred to as the *heavy local traffic assumption*. In this regime, the machines in $\mathcal{M}_l$ cannot accommodate the local arrivals, so we assume $M_r > 0$ to stabilize the system.

Now if λ is in the capacity region, it is easy to see that $\sum_{\vec{L} \in \mathcal{L}} \lambda_{\vec{L}} \leq M_l\alpha + M_r\gamma$. We assume that

$$
\textstyle\sum_{\vec{L} \in \mathcal{L}} \lambda_{\vec{L}} = M_l\alpha + M_r\gamma - \epsilon, \qquad (13)
$$

where $\epsilon > 0$ characterizes the distance between the arrival rate vector and the capacity boundary. The superscript $^{(\epsilon)}$ is used in

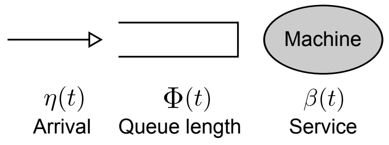


Fig. 2: Lower-Bounding System

this section to indicate the heavy-traffic parameter ϵ. Consider any arrival processes $\left\{A_{\vec{L}}^{(\epsilon)}(t), t \geq 0\right\}_{\vec{L}\in\mathcal{L}}$ with arrival rate vector $\lambda^{(\epsilon)} = \left(\lambda_{\vec{L}_1}^{(\epsilon)}, \lambda_{\vec{L}_2}^{(\epsilon)}, \cdots, \lambda_{\vec{L}_N}^{(\epsilon)}\right)$ satisfying (13). Then

$$\mathbb{E}\Big[\textstyle\sum_{\vec{L}} A_{\vec{L}}^{(\epsilon)}(t)\Big] = \textstyle\sum_{\vec{L}} \lambda_{\vec{L}}^{(\epsilon)} = M_l\alpha + M_r\gamma - \epsilon.$$

The variance of the number of overall arrivals is given by

$$\mathrm{Var}\Big(\textstyle\sum_{\vec{L}} A_{\vec{L}}^{(\epsilon)}(t)\Big) = (\sigma^{(\epsilon)})^2.$$

Denote the queueing and the working status process with such arrival processes as $\left\{\left(Q^{(\epsilon)}(t), f^{(\epsilon)}(t)\right), t \geq 0\right\}$. Later we will let ϵ go to zero to get the *heavy-traffic limit*.

A. Lower Bound

In this subsection, we derive a lower bound on the expectation of the sum of the queue lengths in steady state. Consider a single server queueing system as depicted in Fig. 2. By properly choosing the arrival and the service process, the queue length in this system is stochastically smaller than the sum of the queue lengths in MapReduce. We refer to this single server system as the *lowering bounding system* and the task scheduling system in MapReduce as the *original system*. A lower bound on the expectation of the queue length in steady state in this system is obtained in [12], which is also a lower bound for the original system. Note that this lower bound does not need the heavy local traffic assumption.

Consider the following arrival process $\left\{\eta^{(\epsilon)}(t), t \geq 0\right\}$ and service process $\{\beta(t), t \geq 0\}$:

$$\eta^{(\epsilon)}(t) = \textstyle\sum_{\vec{L}} A_{\vec{L}}^{(\epsilon)}(t),\ \beta(t) = \sum_{i=1}^{M_l} X_i(t) + \sum_{j=1}^{M_r} Y_j(t),$$

where all the processes $\{X_i(t), t \geq 0\}, i = 1, \cdots, M_l$ and $\{Y_j(t), t \geq 0\}, j = 1, \cdots, M_r$ are independent and each process is composed of a sequence of i.i.d. random variables. Let $X_i(t) \sim \text{Bern}(\alpha)$ and $Y_j(t) \sim \text{Bern}(\gamma)$. Then $\mathbb{E}[\beta(t)] = M_l\alpha + M_r\gamma$, and we use ν^2 to denote $\mathrm{Var}(\beta(t))$.

By these settings, the queue length $\Phi^{(\epsilon)}(t)$ in the lower-bounding system is stochastically smaller than the sum of the queue lengths in the original system for any time slot t. Considering the lower bound on $\mathbb{E}\big[\Phi^{(\epsilon)}(t)\big]$ in steady state given by Lemma 4 in [12], we obtain the following theorem.

Theorem 2 (Lower Bound). *For the map task scheduling system in MapReduce, consider any arrival process such that the number of total arrivals at each time slot has expectation $M_l\alpha + M_r\gamma - \epsilon$ and variance $(\sigma^{(\epsilon)})^2$. Suppose the Markov chain $\left\{\left(Q^{(\epsilon)}(t), f^{(\epsilon)}(t)\right), t \geq 0\right\}$ is in steady state under the proposed map-scheduling algorithm. Then, for any t and any ϵ such that $0 < \epsilon < M_l\alpha + M_r\gamma$, the expectation of the sum of the queue lengths in steady state can be lower-bounded as*

$$\mathbb{E}\left[\sum_{m=1}^{M+1} Q_m^{(\epsilon)}(t)\right] \geq \frac{(\sigma^{(\epsilon)})^2 + \nu^2 + \epsilon^2}{2\epsilon} - \frac{M}{2}. \tag{14}$$

Therefore, in the heavy-traffic limit as the arrival rate approaches the service rate from below, assuming the variance $(\sigma^{(\epsilon)})^2$ converges to a constant σ^2, the lower bound becomes

$$\liminf_{\epsilon\to 0^+} \epsilon\mathbb{E}\left[\sum_{m=1}^{M+1} Q_m^{(\epsilon)}(t)\right] \geq \frac{\sigma^2 + \nu^2}{2}. \tag{15}$$

B. State Space Collapse

For a single server queueing system like the one in Fig. 2, the discrete-time Kingman's bound [15] gives an upper bound on the expectation of the queue length in steady state, which is derived by studying the drift of an appropriate Lyapunov function in steady state. The task scheduling system in MapReduce is a more complicated queueing system, which consists of multiple queues and thus has a multi-dimensional state space. However, in the heavy-traffic scenario, we will show that the multi-dimensional state description of the system reduces to a single dimension in the sense that the deviation from a particular direction has bounded moments, independent of the heavy-traffic parameter. This behavior of the queueing system in heavy-traffic scenario is called *state space collapse*. When the state space collapse happens, the system can be analyzed by the similar techniques as used for the single-dimensional system. In this subsection, we will establish the state space collapse for the task scheduling system in MapReduce.

In our model, $\left\{\left(Q^{(\epsilon)}(t), f^{(\epsilon)}(t)\right), t \geq 0\right\}$ is an irreducible, aperiodic, positive recurrent Markov chain with state space $\mathcal{S}$. Since the working status vector f is always finite, we only consider the subspace for the queue lengths. Let $c \in \mathbb{R}_+^{M+1}$ be a vector with unit l_2 norm, then the corresponding parallel and perpendicular components of a queue length vector Q are

$$Q_{\parallel} = \langle Q, c\rangle c, \quad Q_{\perp} = Q - Q_{\parallel}.$$

Throughout this paper, the norm $\|\cdot\|$ refers to l_2 norm. If all the moments of $\|Q_{\perp}\|$, which represent the deviation of the queue length vector from the direction c, are bounded by constants not depending on the heavy-traffic parameter ϵ, we will say that the state space collapses to the direction of c.

Let

$$c = \frac{1}{\sqrt{M_l + 1}}(\underbrace{1, \cdots, 1}_{M_l}, \underbrace{0, \cdots, 0}_{M_r}, 1) \tag{16}$$

be the direction that we will prove the state space collapses to, where the first M_l entries are ones and the following M_r entries are zeros. Consider the Lyapunov function

$$V_{\perp}((Q, f)) = \|Q_{\perp}\|.$$

We can prove that the drift of $V_{\perp}$ satisfies the conditions in Lemma 1 of [12] (see [16] for the derivation of this lemma). Then by this lemma, $V_{\perp}\big((Q^{(\epsilon)}(t), f^{(\epsilon)}(t)\big)$ has bounded moments in steady state, which gives the following theorem.

Theorem 3 (State Space Collapse)**.** *For the map-scheduling system in MapReduce, consider any arrival process with an arrival rate vector strictly within the capacity region satisfying the heavy local traffic assumption, and the number of total arrivals at each time slot has expectation $M_l\alpha + M_r\gamma - \epsilon$ and variance $(\sigma^{(\epsilon)})^2$. Suppose the Markov chain $\left\{\left(Q^{(\epsilon)}(t), f^{(\epsilon)}(t)\right), t \geq 0\right\}$ is in steady state under the proposed map-scheduling algorithm. Then for any t and any ϵ such that*

$$0 < \epsilon < \min\left\{ M_l\alpha + M_r\gamma, M_r\left(M_l+1\right)\gamma, \frac{\left(M_l+1\right)\alpha}{2N}, \frac{\left(M_l+1\right)\min_{\vec{L}\in\mathcal{L}}\lambda_{\vec{L}}}{6} \right\},$$

there exists a sequence of finite numbers $\{C_1, C_2, \cdots\}$ such that for each positive integer r,

$$\mathbb{E}\left[\|Q_{\perp}^{(\epsilon)}(t)\|^r\right] \leq C_r,$$

where the $\perp$ component is w.r.t. the direction defined in (16)*.*

The proof of this theorem is omitted here due to space limit, and available in our technical report [14].

C. Upper Bound and Heavy-Traffic Optimality

In this subsection, we derive an upper bound on the expectation of the sum of the queue lengths in steady state based on the Lyapunov drift-based moment bounding techniques developed in [12], and we show that this upper bound is asymptotically tight under the heavy-traffic regime. The heavy-traffic optimality of joint JSQ and MaxWeight algorithm with homogeneous servers has been established in [13] (in a different context). Due to data locality, our system has heterogeneous servers, which makes the problem more challenging.

We have established the state space collapse for the task scheduling system in MapReduce under the proposed algorithm, so the queue length vector in steady state concentrates along a single direction. Enlightened by the way how the queue length in the single server queueing system is bounded, we treat the multi-dimensional state space in our problem as a one-dimensional state space along the collapse direction and then set the drift of the Lyapunov function $W_{\parallel}\left((Q,f)\right) = \|Q_{\parallel}\|^2$ to zero in steady state to obtain an upper bound for the expected queue length along this direction.

Due to the different service rates in our system, the terms related to service in the Lyapunov drift cannot be bounded directly. We consider the *ideal service process* $\left\{S'(t) = \left(S'_1(t), \cdots, S'_{M+1}(t)\right), t \geq 0\right\}$, which makes the best use of every machine and is defined as

$$S'_m(t) = \begin{cases} X^l_m(t) & \text{if } m \in \mathcal{M}_l \\ 0 & \text{if } m \in \mathcal{M}_r \\ \sum_{m\in\mathcal{M}_r} X^r_m(t) & \text{if } m = M+1 \end{cases}$$

where all the processes $\left\{X^l_m(t), t \geq 0\right\}, m \in \mathcal{M}_l$ and $\left\{X^r_m(t), t \geq 0\right\}, m \in \mathcal{M}_r$ are independent and each process is composed of a sequence of i.i.d. random variables. Let $X^l_m(t) \sim \text{Bern}(\alpha)$ and $X^r_m(t) \sim \text{Bern}(\gamma)$. Utilizing this service process, the queue dynamics (1) can be rewritten as

$$Q(t+1) = Q(t) + A(t) - S'(t) + U'(t), \tag{17}$$

where $U'(t) = S'(t) - S(t) + U(t)$. Since the moments of $S'(t)$ are easy to calculate, we will use this equivalent queue dynamics to express the Lyapunov drift. Then setting the Lyapunov drift to zero gives the following lemma.

Lemma 1. *For the map task scheduling system in MapReduce, consider any arrival process with an arrival rate vector strictly within the capacity region. Suppose the queueing process is in steady state at time slot t under the proposed map-scheduling algorithm, then for any direction c,*

$$\begin{aligned} &2\mathbb{E}\left[\langle c, Q(t)\rangle\langle c, S'(t) - A(t)\rangle\right] \\ &= \mathbb{E}\left[\langle c, A(t) - S'(t)\rangle^2\right] + \mathbb{E}\left[\langle c, U'(t)\rangle^2\right] \qquad (18) \\ &\quad + 2\mathbb{E}\left[\langle c, Q(t) + A(t) - S'(t)\rangle\langle c, U'(t)\rangle\right]. \qquad (19) \end{aligned}$$

The formal proof of this lemma is provided in our technical report [14]. Analyzing each term in this lemma gives the following upper bound, which is asymptotically tight under the heavy-traffic limit. A more detailed proof of the following theorem is also available in our technical report [14].

Theorem 4 (Upper Bound)**.** *For the map-scheduling system in MapReduce, consider any arrival process with an arrival rate vector strictly within the capacity region satisfying the heavy local traffic assumption, and the number of total arrivals at each time slot has expectation $M_l\alpha + M_r\gamma - \epsilon$ and variance $(\sigma^{(\epsilon)})^2$. Suppose the Markov chain $\left\{\left(Q^{(\epsilon)}(t), f^{(\epsilon)}(t)\right), t \geq 0\right\}$ is in steady state under the proposed map-scheduling algorithm. Then for any t and any ϵ such that*

$$0 < \epsilon < \min\left\{ M_l\alpha + M_r\gamma, M_r\left(M_l+1\right)\gamma, \frac{\left(M_l+1\right)\alpha}{2N}, \frac{\left(M_l+1\right)\min_{\vec{L}\in\mathcal{L}}\lambda_{\vec{L}}}{6} \right\}, \tag{20}$$

the expectation of the sum of the queue lengths in steady state can be upper bounded as

$$\mathbb{E}\left[\sum_{m=1}^{M+1} Q^{(\epsilon)}_m(t)\right] \leq \frac{(\sigma^{(\epsilon)})^2 + \nu^2}{2\epsilon} + B^{(\epsilon)}, \tag{21}$$

where $B^{(\epsilon)} = o(\frac{1}{\epsilon})$, i.e., $\lim\limits_{\epsilon\to 0^+} \epsilon B^{(\epsilon)} = 0$.

Therefore, in the heavy-traffic limit as the arrival rate approaches the service rate from below, assuming the variance $(\sigma^{(\epsilon)})^2$ converges to a constant σ^2 the upper bound becomes

$$\limsup_{\epsilon\to 0^+} \epsilon\mathbb{E}\left[\sum_{m=1}^{M+1} Q^{(\epsilon)}_m(t)\right] \leq \frac{\sigma^2 + \nu^2}{2}. \tag{22}$$

This upper bound under heavy-traffic limit coincides with the lower bound (15)*, which establishes the first moment heavy-traffic optimality of the proposed algorithm.*

Proof: Fix an ϵ that satisfies (20) and then we temporarily omit the superscript $^{(\epsilon)}$ for simplicity. Since we will study the performance in steady state, we assume that the Markov chain $\{(Q(t), f(t)), t \geq 0\}$ is in steady state from time slot 0 and consider the equation in Lemma 1 for any $t \geq 0$ with the collapse direction c defined in (16).

First, by the definition of $S'(t)$ and the property of steady state, the term on the left side of (18) satisfies

$$\mathbb{E}\left[\langle c, Q(t)\rangle\langle c, S'(t) - A(t)\rangle\right] = \frac{\epsilon}{M_l + 1}\mathbb{E}\left[\sum_{m=1}^{M+1} Q_m(t)\right].$$

Next, we study the two terms on the right side in (18). Recall the definition of ν^2 in the lower-bounding system. Then

$$\mathbb{E}\left[\langle c, A(t) - S'(t)\rangle^2\right] = \frac{1}{M_l + 1}\left((\sigma^{(\epsilon)})^2 + \nu^2 + \epsilon^2\right).$$

For the other term $\mathbb{E}\left[\langle c, U'(t)\rangle^2\right]$, since $Q(t)$ is in steady state,

$$\mathbb{E}\left[\langle c, A(t) - S'(t) + U'(t)\rangle\right] = \mathbb{E}\left[\langle c, Q(t+1) - Q(t)\rangle\right] = 0.$$

Therefore

$$\mathbb{E}\left[\langle c, U'(t)\rangle\right] = \mathbb{E}\left[\langle c, S'(t) - A(t)\rangle\right] = \frac{\epsilon}{\sqrt{M_l + 1}}$$

$$\mathbb{E}\left[\langle c, U'(t)\rangle^2\right] \leq \frac{M}{\sqrt{M_l + 1}}\mathbb{E}\left[\langle c, U'(t)\rangle\right] = \frac{\epsilon M}{M_l + 1}.$$

Finally we bound the term (19). To bound the expectation of $\langle c, Q(t)\rangle\langle c, U'(t)\rangle$, we write it as

$$\begin{aligned}\langle c, Q(t)\rangle\langle c, U'(t)\rangle &= \langle Q(t), U'(t)\rangle - \langle Q_\perp(t), U'_\perp(t)\rangle \\ &= \langle Q(t), S'(t) - S(t)\rangle + \langle Q(t), U(t)\rangle - \langle Q_\perp(t), U'_\perp(t)\rangle.\end{aligned}$$

In the case that the service time is deterministic, $Q(t)$ and $U'(t)$ are orthogonal, so we can directly apply the state space collapse result to bound $\langle Q_\perp(t), U'_\perp(t)\rangle$. However, the service time in our model is random. To bound $\langle Q(t), U'(t)\rangle$ with a small number, we start from the following inequality,

$$\mathbb{E}\left[\langle Q(t), S'(t) - S(t)\rangle\right] \leq R_1\sqrt{M_l + 1}\mathbb{E}\left[\langle c, S'(t) - S(t)\rangle\right],$$

where $R_1 > 0$ is a constant. We sketch the proof of this inequality as follows and refer to Lemma 17 in our technical report [14] for details. First we expand the inner product and see that it is sufficient to show that for any $m \in \mathcal{M}_l$,

$$\begin{aligned}&\mathbb{E}\left[Q_m(t)\left(X^l_m(t) - S^l_m(t)\right) + \overline{Q}(t)\left(-S^r_m(t)\right)\right] \\ &\leq R_1\mathbb{E}\left[X^l_m(t) - S^l_m(t)\right].\end{aligned}$$

Then we use $\big(Q(t), f(t)\big)$ to decompose the probability space. For the case $f_m(t) = 0$, machine m makes a scheduling decision at the current time slot, so the inequality follows from the MaxWeight policy. The proof for the case $f_m(t) = 1$ is straightforward since the actual service has the same distribution as the ideal service. For the case $f_m(t) = 2$, we still consider τ^t_m defined in (11). Decomposing the probability space further by the random variable τ^t_m and utilizing the MaxWeight policy and the boundedness of arrival and service yield

$$\begin{aligned}&\mathbb{E}\Big[Q_m(t)\left(X^l_m(t) - S^l_m(t)\right) + \overline{Q}(t)\left(-S^r_m(t)\right) \\ &\quad \Big|\tau^t_m = t - n, Q(t) = Q, f(t) = f\Big] \leq n\left(NA_{\max}\alpha + M\gamma\right).\end{aligned}$$

By the geometric distribution of the service time, the probability $\Pr\left(\tau^t_m = t - n \mid Q(t) = Q, f(t) = f\right)$ is proportional to $(1-\gamma)^{n-1}$. Thus the conditional expectation over the subspace $f_m(t) = 2$ is bounded by a constant times $\sum_{n=1}^{\infty} n(1-\gamma)^{n-1}$, which is also bounded by a constant, and then the inequality follows for proper coefficient R_1. Integrating these three cases gives the inequality we want to prove. This inequality indicates that the MaxWeight algorithm results in a small difference between the actual service $S(t)$ and the ideal service $S'(t)$ in the sense that the queue length vector $Q(t)$ has finite projection in the direction $S'(t) - S(t)$ on average. Now by definitions,

$$\langle Q(t), U(t)\rangle = \overline{Q}(t)\overline{U}(t) \leq M\overline{U}(t) \leq M\sqrt{M_l + 1}\langle c, U(t)\rangle.$$

Let $R_2 = \max\{R_1, M\} > 0$, then

$$\begin{aligned}&\mathbb{E}\left[\langle Q(t), S'(t) - S(t)\rangle + \langle Q(t), U(t)\rangle\right] \\ &\leq R_2\sqrt{M_l + 1}\mathbb{E}\left[\langle c, S'(t) - S(t)\rangle + \langle c, U(t)\rangle\right] \\ &= R_2\sqrt{M_l + 1}\mathbb{E}\left[\langle c, U'(t)\rangle\right] = \epsilon R_2.\end{aligned}$$

To bound the term $-\langle Q_\perp(t), U'_\perp(t)\rangle$, we use the state space collapse theorem, which claims that there exists a constant C_2 such that $\mathbb{E}\left[\|Q_\perp(t)\|^2\right] \leq C_2$. Then using the Cauchy-Schwartz inequality and this bound yields,

$$\begin{aligned}\mathbb{E}\left[-\langle Q_\perp(t), U'_\perp(t)\rangle\right] &\leq \sqrt{\mathbb{E}\left[\|Q_\perp(t)\|^2\right]\mathbb{E}\left[\|U'_\perp(t)\|^2\right]} \\ &\leq \sqrt{C_2\mathbb{E}\left[\|U'(t)\|^2\right]} \leq \sqrt{\epsilon C_2 M}.\end{aligned}$$

Meanwhile, the number of arrivals in each time slot is bounded. Thus the term (19) can be bounded as

$$\begin{aligned}&\mathbb{E}\left[\langle c, Q(t) + A(t) - S'(t)\rangle\langle c, U'(t)\rangle\right] \\ &\leq \epsilon\left(R_2 + \frac{NA_{\max}}{M_l + 1}\right) + \sqrt{\epsilon C_2 M}.\end{aligned}$$

We revive the superscript $^{(\epsilon)}$ now. Combining the inequalities for the terms in the equation in Lemma 1 yields

$$\mathbb{E}\left[\sum_{m=1}^{M+1} Q^{(\epsilon)}_m(t)\right] \leq \frac{(\sigma^{(\epsilon)})^2 + \nu^2}{2\epsilon} + B^{(\epsilon)},$$

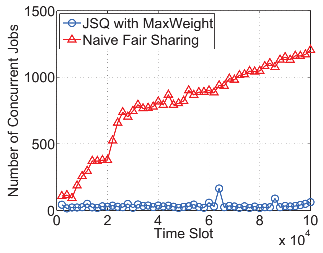
where

$$B^{(\epsilon)} = \frac{\epsilon}{2} + \frac{M}{2} + (M_l + 1)R_2 + NA_{\max} + (M_l + 1)\sqrt{\frac{C_2 M}{\epsilon}}.$$

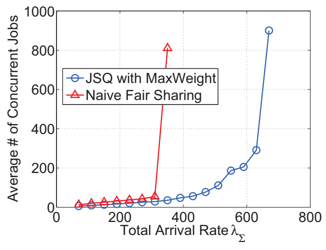
Obviously $\lim_{\epsilon \to 0^+} \epsilon B^{(\epsilon)} = 0$, thus $B^{(\epsilon)} = o\left(\frac{1}{\epsilon}\right)$. Then the bound (22) for the heavy-traffic limit follows immediately by taking limits on both sides. ■

V. Simulations

In this section, we use simulations to compare the throughput and delay performance of the proposed algorithm with the naïve fair sharing algorithm proposed in [7]. The naïve fair sharing shows a great performance improvement over the Hadoop's FIFO scheduler according to the evaluation in [7]. The related simulation parameters are from mimicking real workload analyzed in [17].

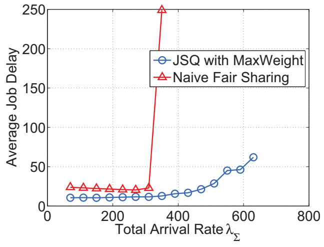


(a) Total Arrival Rate $\lambda_\Sigma = 390$

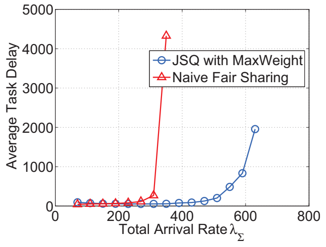


(b) Throughput Region

Fig. 3: Throughput Performance



(a) Job Delay in Steady State



(b) Task Delay in Steady State

Fig. 4: Delay Performance.

We consider a computing cluster with 1000 machines and a dataset distributed uniformly in 800 of the them. The service rates for local and remote tasks are $\alpha = 0.8$ and $\gamma = 0.2$, respectively, so we only consider the total task arrival rate λ_Σ less than $800\alpha + 200\gamma = 680$ per time slot. For each λ_Σ we simulate the system for one sample path. As noted in Section II, we maintain multiple subqueues for each queue, and the subqueue corresponding to the job with the fewest running tasks is selected during scheduling, as in the naïve fair sharing.

Throughput Performance. We keep track of the number of concurrent jobs in the system to observe stability. Fig. 3a shows a representative sample of the evolution of this number over time, indicating the comparison of instability and stability. Fig. 3b shows the average number of concurrent jobs over the last $250,000$ time slots for each λ_Σ. The turning points at 350 and 630 indicate the throughput difference. From these results we can conjecture that the proposed algorithm achieves the maximum throughput, and the throughput is increased by more than 80% compared with the naïve fair sharing.

Delay Performance. For each λ_Σ, we calculate the average delay for jobs and tasks departing in steady state and illustrate the results in Fig. 4. We did not plot the results for $\lambda_\Sigma \geq 390$ under the naïve fair sharing and for $\lambda_\Sigma = 670$ under the proposed algorithm since the delay becomes very large (more than ten times larger) due to instability, which also confirms the throughput difference of the two algorithms. For small arrival rates, the proposed algorithm roughly halves the average job delay compared with the naïve fair sharing (Fig. 4a), while the average task delay are roughly the same (Fig. 4b).

VI. Conclusion

We considered map scheduling algorithms in MapReduce with data locality. We first presented the capacity region of a MapReduce computing cluster with data locality and then we proved the throughput optimality. Beyond throughput, we showed that the proposed algorithm asymptotically minimizes the number of backlogged tasks as the arrival rate vector approaches the boundary of the capacity region, i.e., it is heavy-traffic optimal.

Acknowledgement

Research supported in part by NSF Grants ECCS-1255425.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *ACM Commun.*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[2] "Hadoop," http://hadoop.apache.org.

[3] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, "Scarlett: coping with skewed content popularity in mapreduce clusters," in *Proc. European Conf. Computer Systems (EuroSys)*, Salzburg, Austria, 2011, pp. 287–300.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Proc. ACM Symp. Operating Systems Principles (SOSP)*, Bolton Landing, NY, 2003, pp. 29–43.

[5] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *IEEE Symp. Mass Storage Systems and Technologies (MSST)*, Incline Villiage, NV, May 2010, pp. 1–10.

[6] T. White, *Hadoop: The definitive guide*. Yahoo Press, 2010.

[7] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *Proc. European Conf. Computer Systems (EuroSys)*, Paris, France, 2010, pp. 265–278.

[8] C. Abad, Y. Lu, and R. Campbell, "DARE: Adaptive data replication for efficient cluster scheduling," in *IEEE Int. Conf. Cluster Computing (CLUSTER)*, Austin, TX, 2011, pp. 159–168.

[9] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An analysis of traces from a production mapreduce cluster," in *Proc. IEEE/ACM Int. Conf. Cluster, Cloud and Grid Computing (CCGRID)*, Melbourne, Australia, 2010, pp. 94–103.

[10] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 4, pp. 1936–1948, Dec. 1992.

[11] S. T. Maguluri and R. Srikant, "Scheduling jobs with unknown duration in clouds," in *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, Turin, Italy, 2013.

[12] A. Eryilmaz and R. Srikant, "Asymptotically tight steady-state queue length bounds implied by drift conditions," *Queueing Syst.*, vol. 72, no. 3-4, pp. 311–359, Dec. 2012.

[13] S. T. Maguluri, R. Srikant, and L. Ying, "Heavy traffic optimal resource allocation algorithms for cloud computing clusters," in *Int. Teletraffic Congr. (ITC)*, Krakow, Poland, 2012.

[14] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality," Arizona State Univ., Tempe, AZ, Tech. Rep., Jul. 2012.

[15] J. F. C. Kingman, "Some inequalities for the queue GI/G/1," *Biometrika*, vol. 49, no. 3-4, pp. 315–324, Dec. 1962.

[16] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Ann. Appl. Prob.*, pp. 502–525, 1982.

[17] G. Ananthanarayanan, A. Ghodsi, A. Wang, D. Borthakur, S. Kandula, S. Shenker, and I. Stoica, "Pacman: coordinated memory caching for parallel jobs," in *Proc. Conf. Networked Systems Design and Implementations (USENIX)*, 2012, pp. 20–20.